

LA-UR-17-20796

Approved for public release; distribution is unlimited.

Title: Codesign Performance Prediction for Computational Physics 3rd Year
Review Overview talk

Author(s): Eidenbenz, Stephan Johannes
Zerr, Robert Joseph

Intended for: Report

Issued: 2017-02-02

Disclaimer:

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Codesign Performance Prediction for Computational Physics 3rd Year Review Overview talk

January 26, 2017

Stephan Eidenbenz, Joe Zerr
Los Alamos National Laboratory (LANL)

Performance Prediction Team FY17

Stephan Eidenbenz (PI)	NSEC	Joe Zerr (co-PI)	CCS-2
Olena Tkachenko (post-BAC)	NSEC	Balu Nadiga	CCS-2
		Max Rosa	CCS-2
Nandkishore Santhi (co-PI)	CCS-3		
Guillaume Chapuis (PD)	CCS-3	Rick Zamora	T-1
Gopinath Chennupathi	CCS-3		
Sunil Thulasidasan	CCS-3	Patricia Grubel (PD)	CCS-7
Jason Liu	Florida Intl U		
Ahmed Kishwar (GRA)	Florida Intl U		

Table of Contents

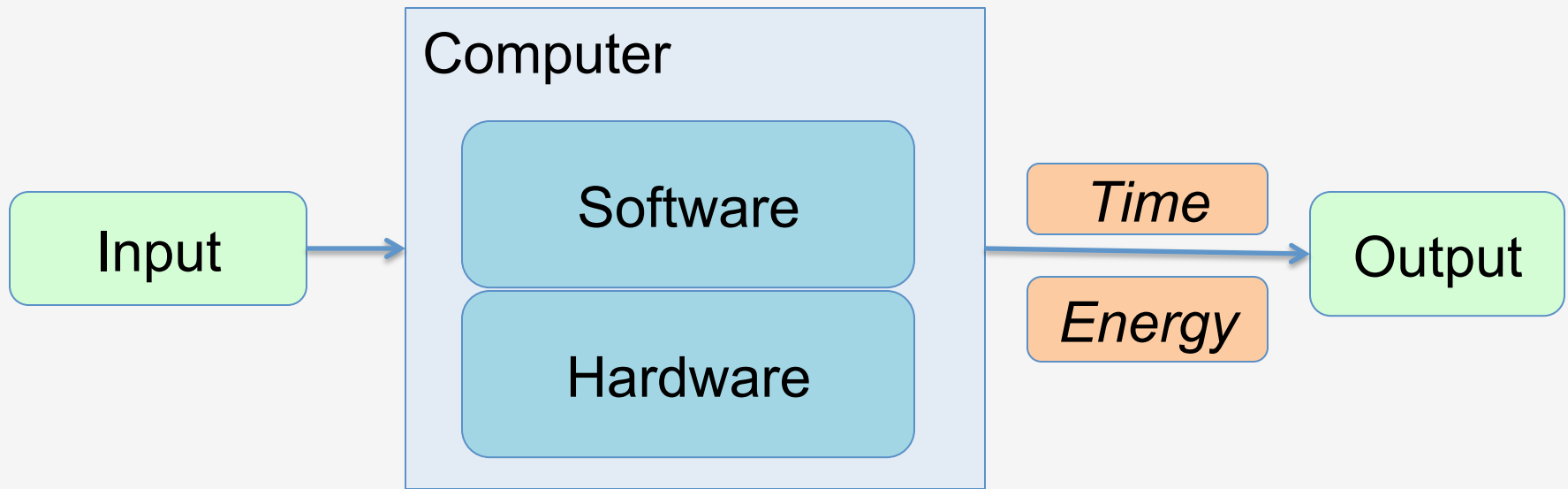
- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
- Middleware Models
- Hardware Models
- Parallel Discrete Event Simulation – Simian
- List of Publications, Presentations, Software
- Outlook

Our problem: Coping with Novel Architectures to Optimize Software Performance

- End of hardware scaling laws around 2005 has led to novel hardware architectures
 - Multi-core, many-core
 - Accelerator techniques: Vector units, Graphics Processing Units (GPU)
 - Pipelines, Prefetching, Speculative execution
- Hardware changes disruptive to performance of existing software code base. Require complex software changes by high-skill software architects/computational physicists
 - Parallelism: distributed-memory, shared-memory, instruction-level
 - Latency-hiding, data movement/motion, fault resilience
- Traditional coping strategies
 - Software engineer skills improvement programs
 - Middle-ware libraries
 - Code instrumentation, mini-apps

Our Solution: Codesign Modeling to *Predict* Performance of Novel SW/Computational Methods on Novel HW Platforms

Model of “Computing”

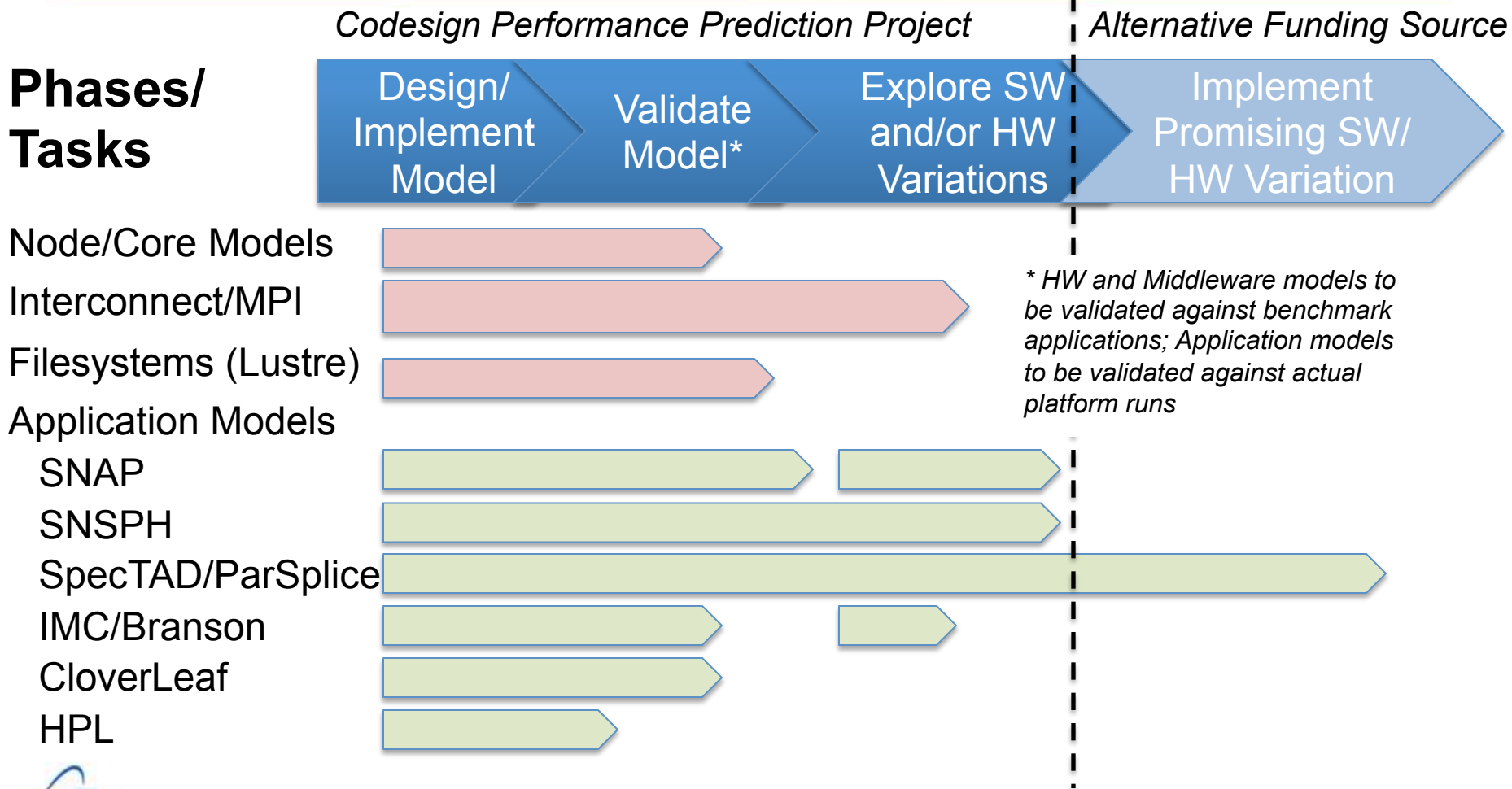


Key idea: Explore (Parameterized) SW and HW Design Spaces and Assess Algorithmic Variations

 ***Predicted***
Performance Measures

 *Design Spaces*

Project Phases (and Status January 2017): Model Validation before Design Space Exploration



Performance Prediction Toolkit (PPT): Jan 2017

Rapid Prototyping Modeling (Python or Lua): Simple, Modular

Codes

Hardware Models

- Parameterized Model Hierarchy (Clusters, Nodes, Cores)
e.g., Mustang, Trinity, Cielo, Titan, Moonlight, AMD Opteron, KNL, MacPro
- Parameterized Memory hierarchy, Pipeline Models for input (data dependency) graphs
- Accelerators
K20X, K40, K6000, M2090, Pascal
- Interconnect
Torus
FatTree
Butterfly

Middle-ware Models

- MPI
- OpenMP
- *Legion*

Application Models

- | | |
|----------------------|-----------------------|
| • SNAPSim | Det. Transport |
| • SPHSim | Hydrodynamics |
| • IMCSim | Monte Carlo Transport |
| • MDSim | MD |
| • SpecTADSim | MD |
| • CWBJ “Jacobi” | Det. Transport |
| • CloverLeafSim | Hydrodynamics |
| • HPLSim (“Linpack”) | Benchmark |
| • HPCGSim | Benchmark |
| • PolyBenchSim | Benchmark |
| • ParboilSim GPU | Benchmark |
| • Misc | |

Simian – Parallel Discrete Simulation Engine

PPT: Hardware Modeling Approach

1. Define what hardware resource is modeled (e.g., core)
2. Define and set Hardware Parameters
 - Example: Clock speed, cycles per ALU operation, cycles per RAM access, L2-cache size, ...
 - Set parameter values (“First-principles”: according to (anticipated) spec sheets, or fitted to data)
3. Implement high-level instruction API to application/SW model (“tasklist”)
 - Example: tasklist = [23 mem accesses, 55 float ops, 30 vec_ops]
 - Function “compute(tasklist)” calculates the time it takes to execute tasklist
 - NEW ALTERNATIVE: More complex basic-block-wise cycle-accurate TASK GRAPHLETS as input with data dependencies among code instruction (Talk by N. Santhi)

PPT: Software Modeling Approach

1. Model the loop structure of the code within a pseudo-code-like Simian Process, usually using MPI model, usually without computing the physics
2. Identify time-intensive inner loops to synthesize into API instructions (“tasklists”) for hardware model
 - Expert opinion
 - Code profiling
3. Synthesize inner loops (create “tasklists” or “task graphs”) through
 - Manual method/code analysis
 - Runtime analyses tools (architecture independent ByFL)
 - *NEW: Automated analysis on LLVM level for task-graph formation*

PPT is a tool (and design philosophy) mainly for Application Developers, Code Teams, and perhaps Middleware Developers. Such a focus is unique among performance prediction tools

Table of Contents

- Introduction: Performance Prediction Toolkit (PPT)



- Application Models
- Middleware Models
- Hardware Models
- Parallel Discrete Event Simulation – Simian
- List of Publications, Presentations
- Outlook

Application Models

- SPHSim

- IMCSim

*Result Overview
(Stephan)*

- SNAPSim

- CWBJSim

- CloverSim

*More detailed
presentation
(Joe)*

- HPLSim

- MD-related simulators

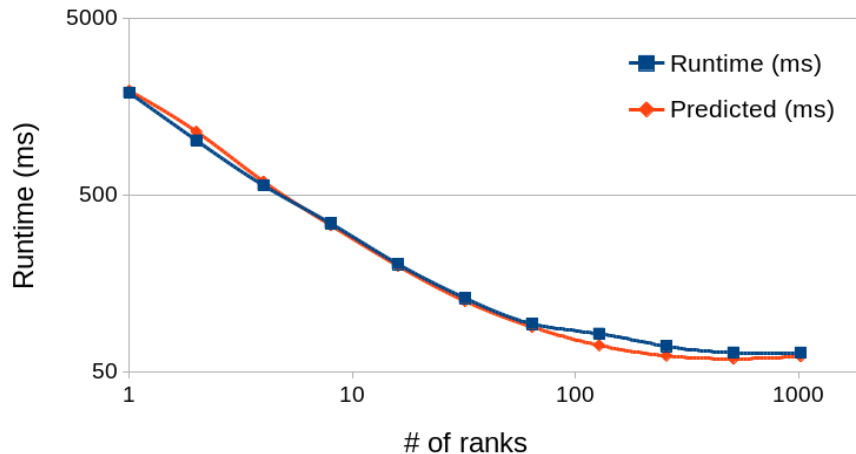
- MDSim, ParSplice, SpecTADSim

*Separate talk
(Rick Zamora)*

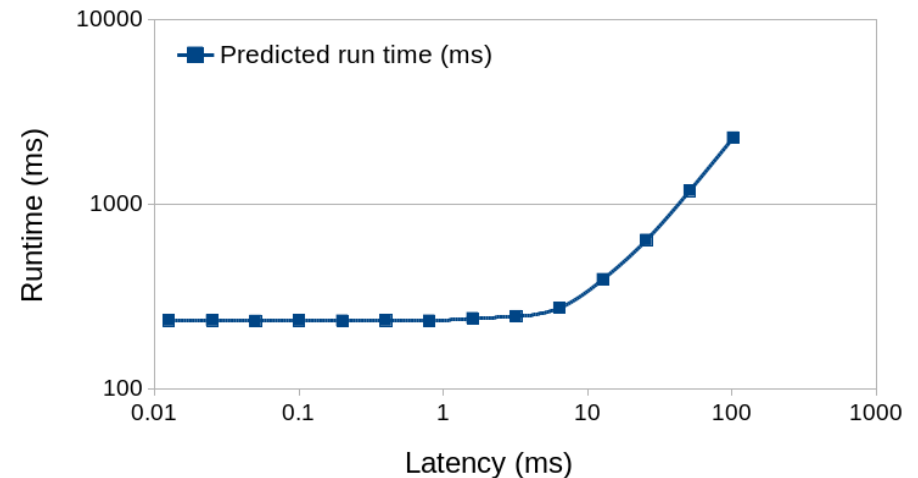
Smoothed Particle Hydrodynamics: SPHSim

[G. Chapuis et al., WinterSim 2016]

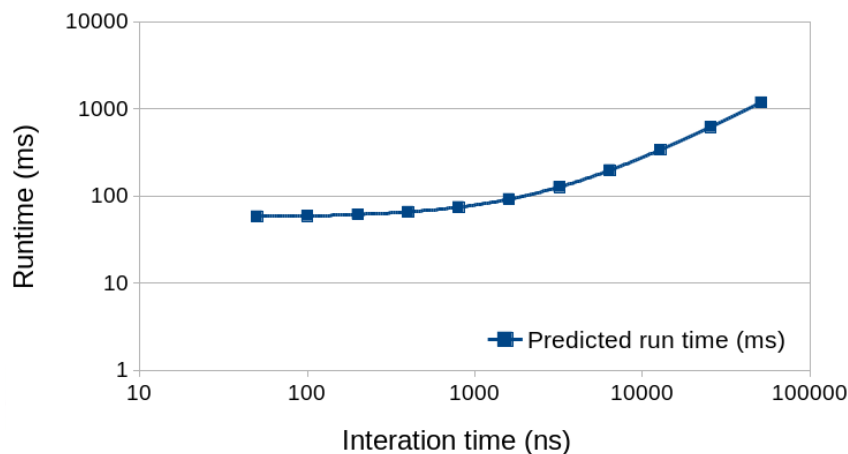
Runtime predictions



Effect of latency



Effect of interaction time



- *SNSPH*: Compute distance-dependent particle interactions through hierarchical Oct-Tree data structure as a latency-hiding mechanisms
- *SPHSim*: *stochastic application model in Simian Lua*
- Validate, Parameter study: no need for lower latency interconnect, physics kernel opt still useful

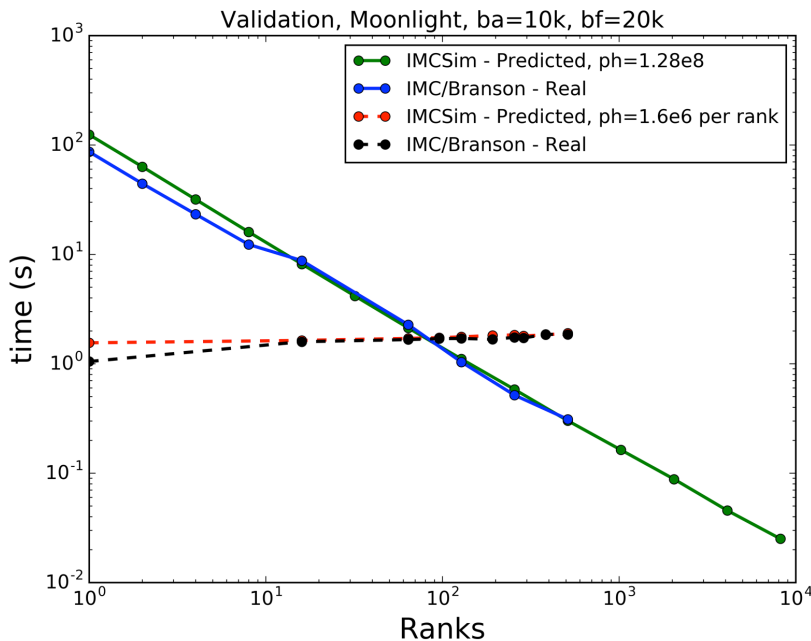
CLASSIFIED

Slide 12

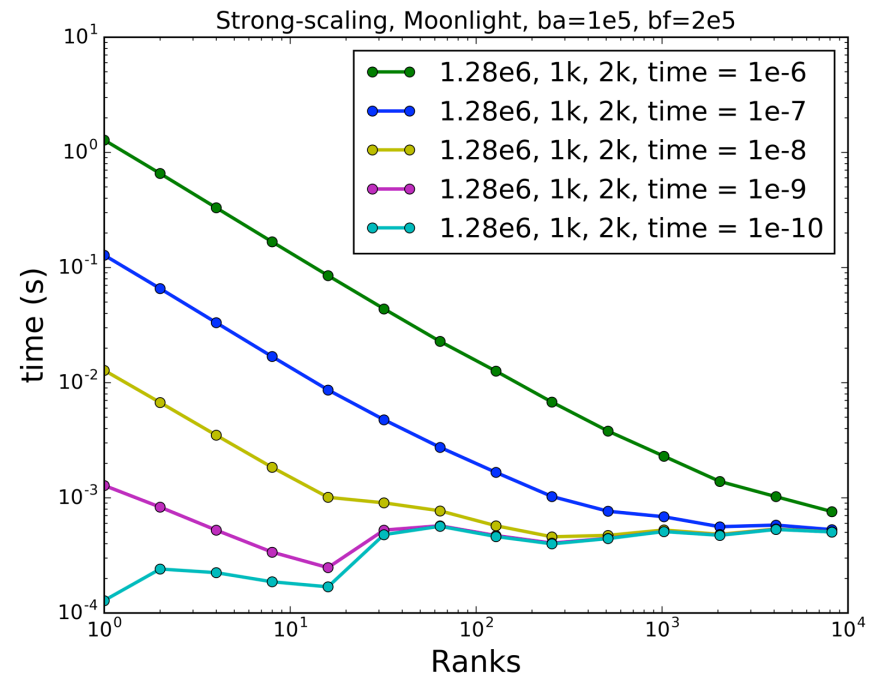
Implicit Monte Carlo: IMCSim

[In preparation]

Validation: Strong scaling and Weak Scaling



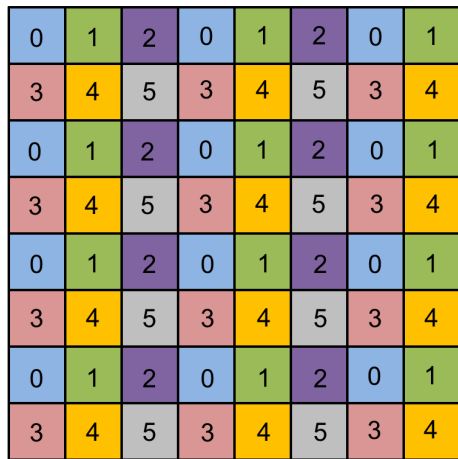
Effect of Compute Kernel Time



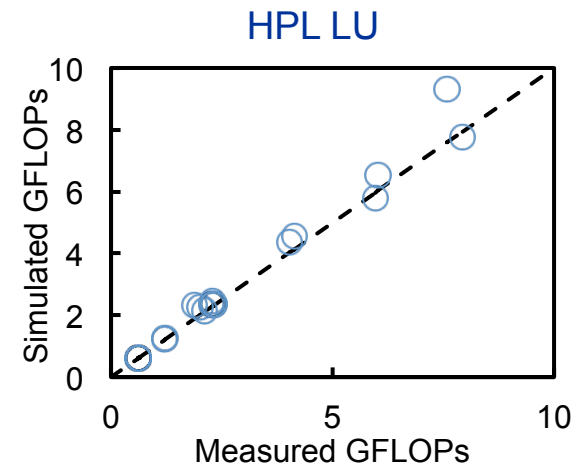
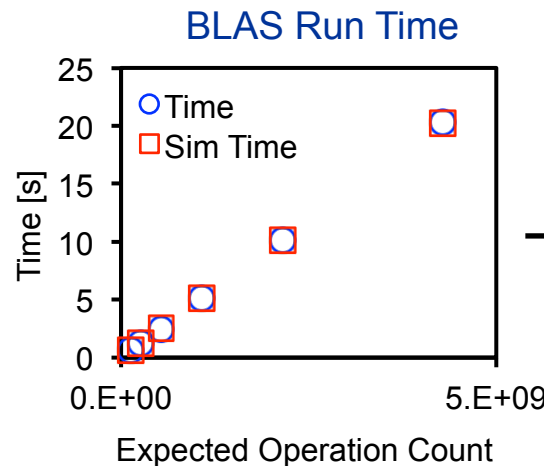
- Implicit Monte Carlo (IMC) and the Branson mini-app are an example of non-deterministic radiation transport codes
- IMCSim enables us to study computation vs. communication trade-offs and identifies optimal MPI parameter settings

High-Performance Linpack (HPL) - *HPL-Sim*

- PPT Parallel Linpack Prediction under development
 - **HPL-Sim** = HPL Algorithm + BLAS runtime estimates + PPT



+



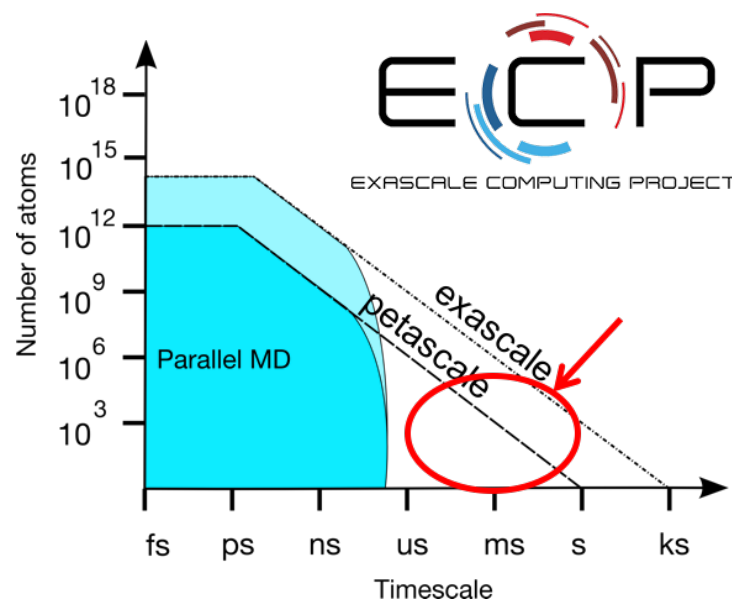
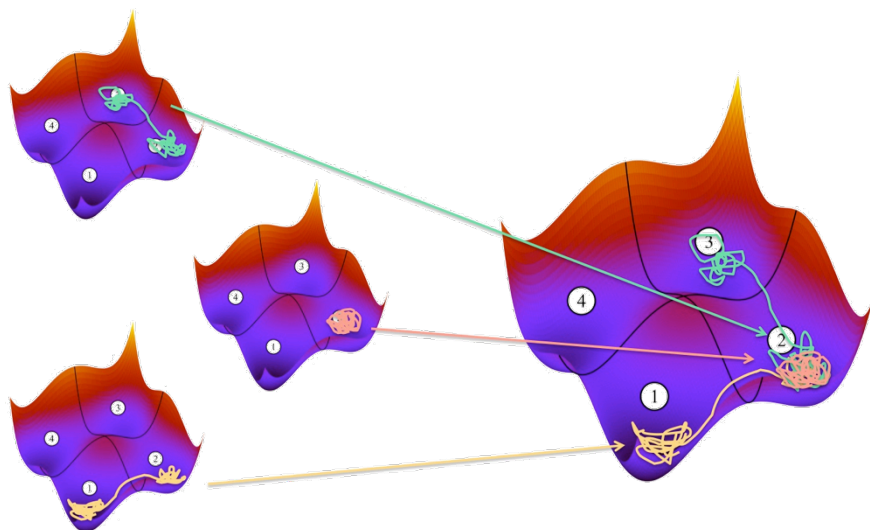
Motivation:
(top500.org)

TOP 10 Sites for November 2016

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
3	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209

Exascale Molecular Dynamics

- PPT is being leveraged by both **BES** and **ECP** projects to develop and plan the implementation of parallel Accelerated Molecular Dynamics (AMD)
 - **BES**: Temperature-Accelerated Parallel Splicing (TAPS)
 - **ECP**: Exascale Atomistic for Accuracy Length and Time (EXAALT)



SNAPSim

Developing SNAPSim for linear, deterministic transport applications from SNAP

- Deterministic transport for a structured grid of spatial “cells”, solving for a set of **nang** discrete directions of particle travel, moving at speeds within **ng** bins of energy “groups”
- Outer/inner iteration strategy to resolve group-to-group interactions (outer) and within group interactions (inner)
 - Outer is a large matrix-vector multiplication
 - Inner is a transport *mesh sweep* that updates solution guess for all cells, directions, groups in a *highly ordered manner*
- “Optimal” performance of sweeps is a long-studied computational science problem: requires mix of parallel strategies, balance of intranode and internode tradeoffs

SNAPSim overview

- Execution time dominated by mesh sweeps and group-to-group calculations
 - *A priori* knowledge of task graph and workload per task
 - Fixed placement of tasks on cores → entities
 - tasklist items that are relatively predictable

```
Call time_compute with work chunk tasklist = [integer, float, vector, and memory ops]
```

```
Do a single sweep for each energy group:
```

```
    each process determines requirements
```

```
    wait until requirements are satisfied (with simulated MPI messages)
```

```
    advance simulation time with time_compute result
```

```
    determine dependents and send messages to inform chunks are complete
```

```
Call time_compute for outer source
```

```
For timestep_n:
```

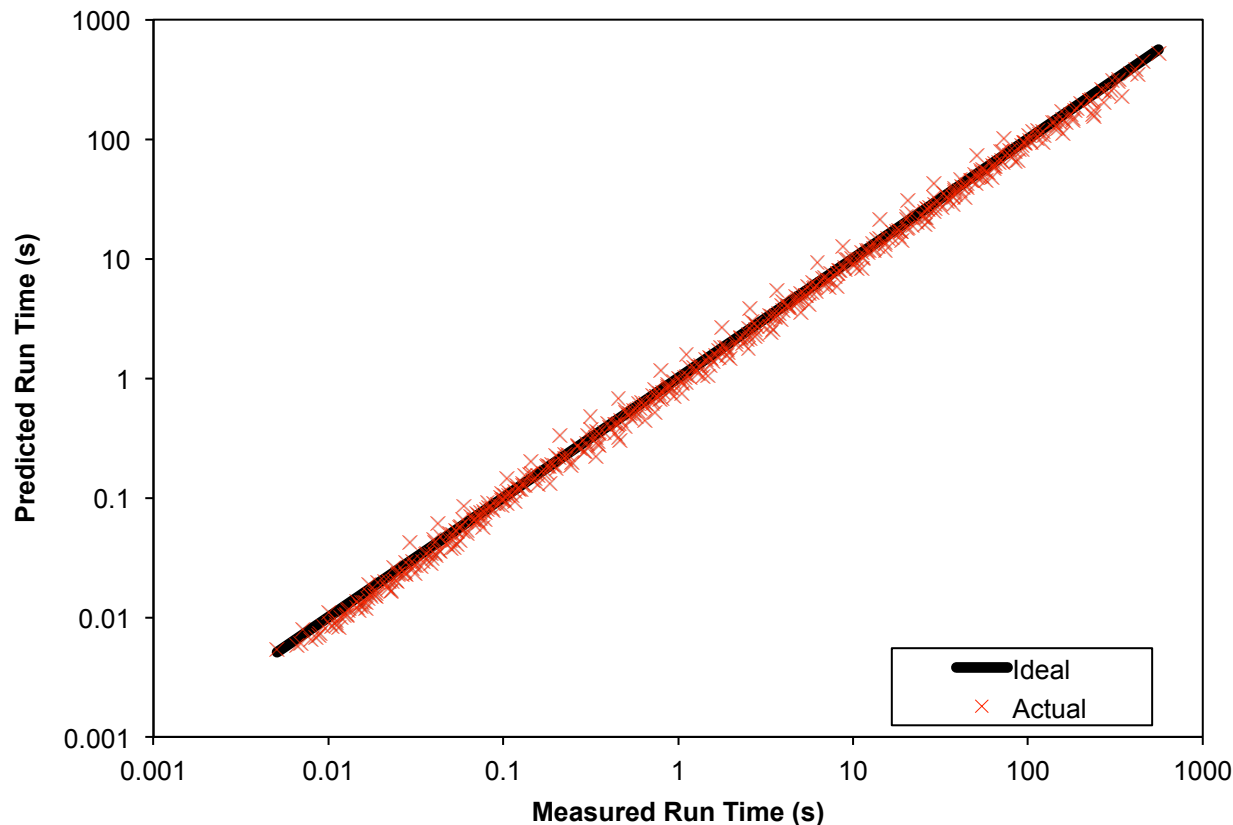
```
    for outer_o:
```

```
        for inner_i:
```

```
            accumulate time from single sweep
```

```
            accumulate time from single outer source calculation
```

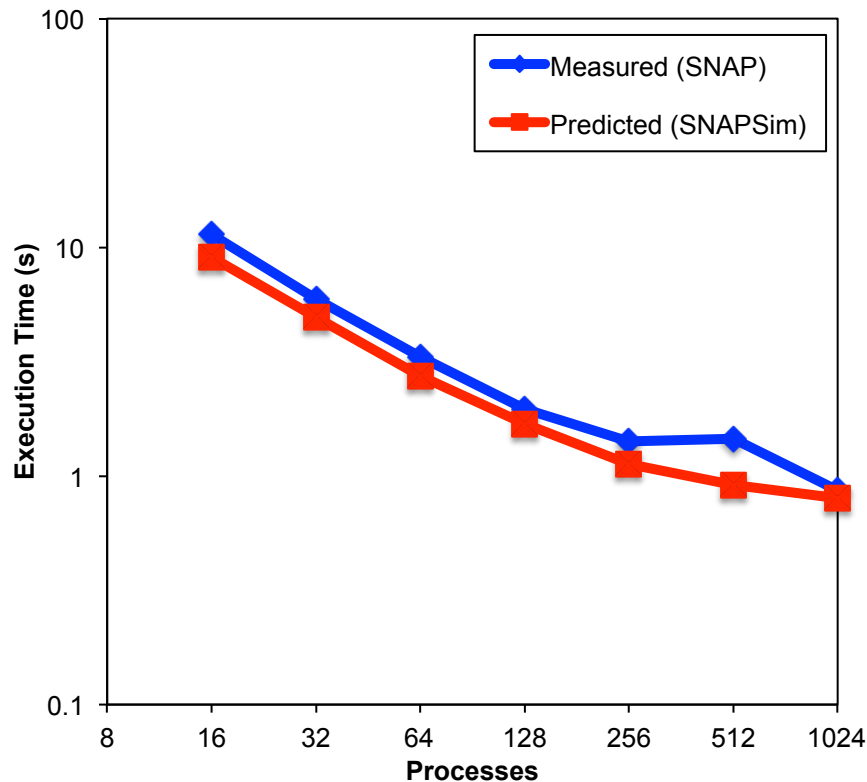
Serial validation testing suite on Moonlight: 500 jobs varying physical domain size



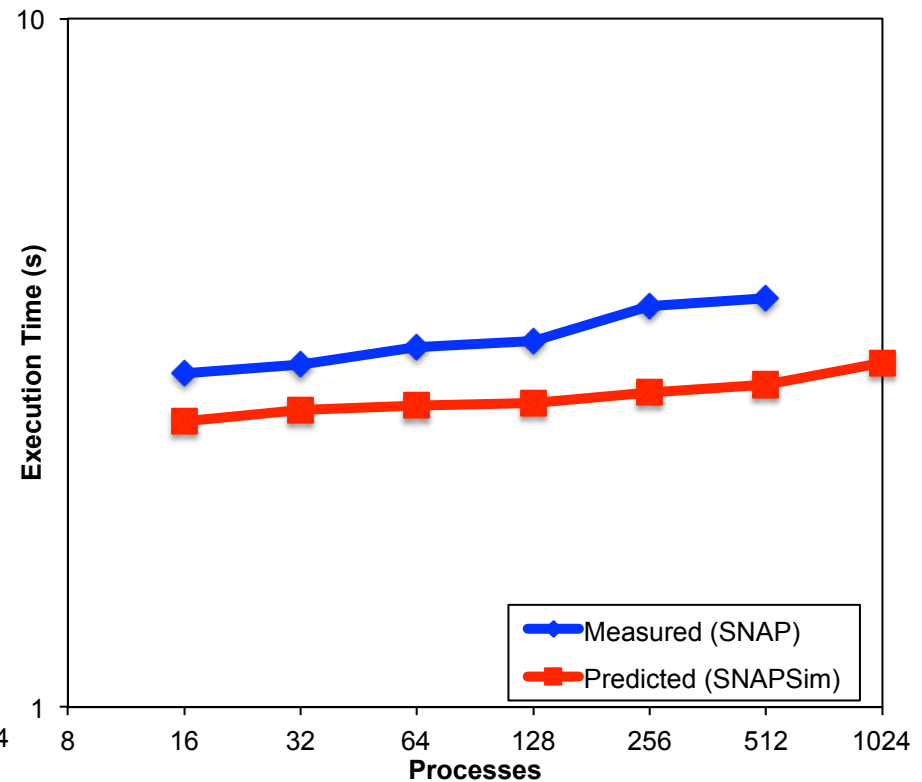
- Use a HW model with developer estimates of cache efficiency
- Relative differences typically within 10%
- Expected under-prediction because simulator does not include everything
- Worst comparisons often represent less important cases

Moonlight strong and weak scaling tests with full interconnect and MPI models

Strong (cells=65k, nang=48, ng=42)



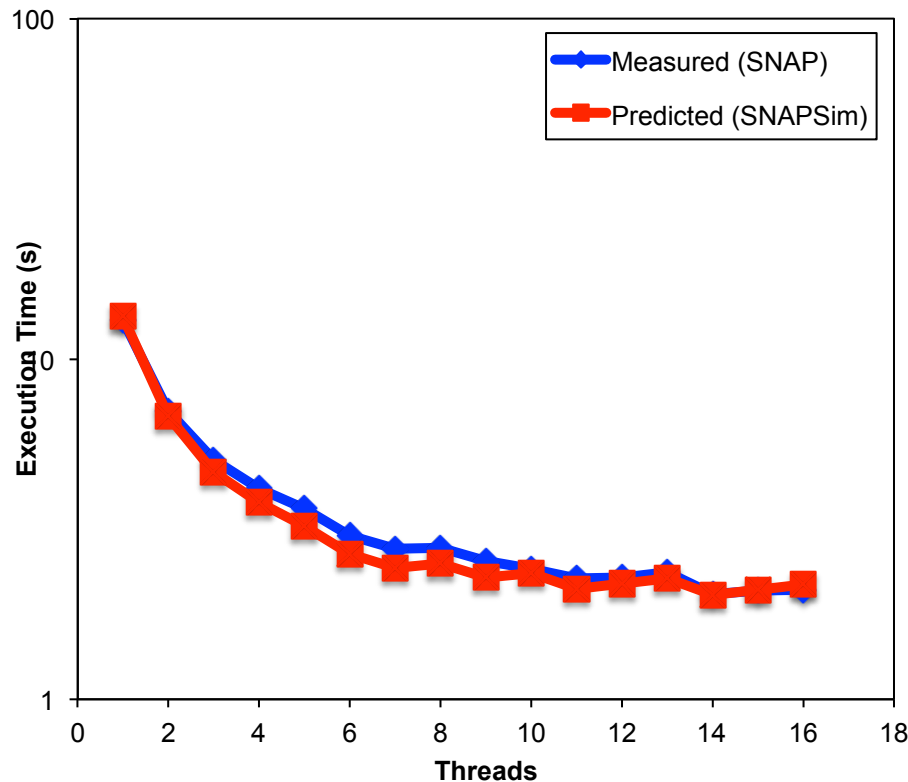
Weak (cells=16k/rank, nang=48, ng=42)



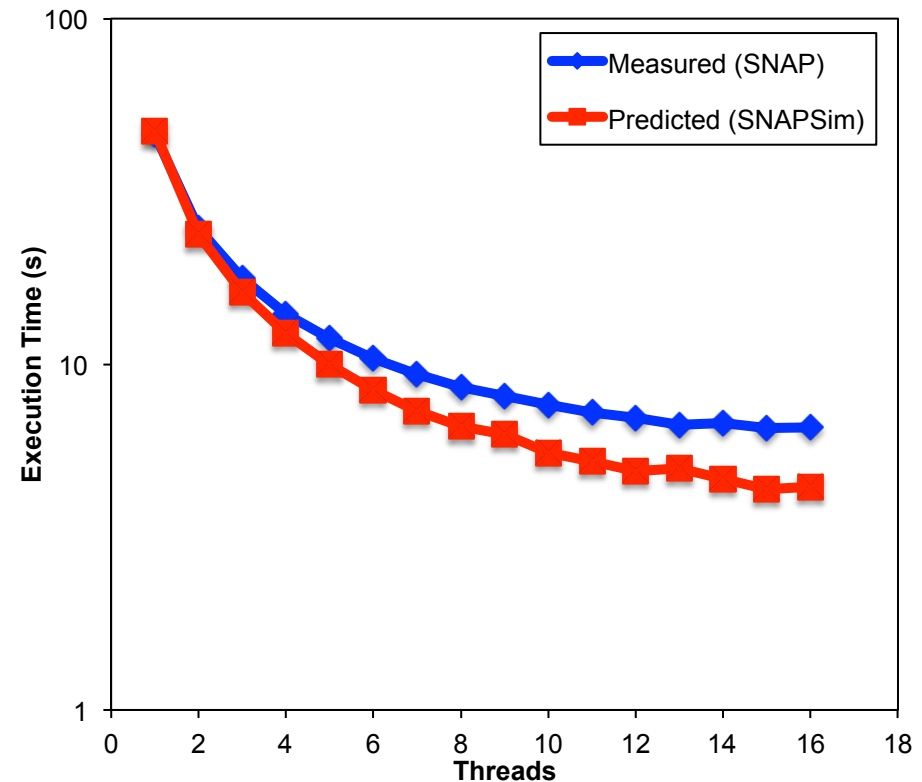
Captures general trends. Larger problem necessary for weak scaling to observe more interesting trends associated with algorithm.

SNAPSim with threading middleware applied to parallelize work over energy groups

Strong (ng=42)



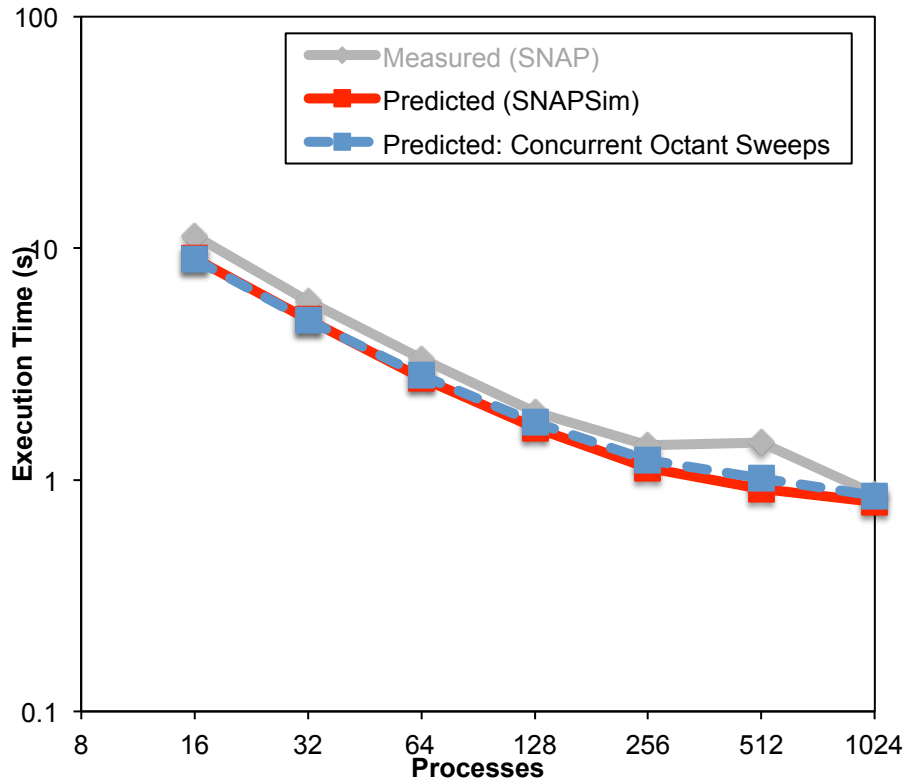
Strong (ng=118)



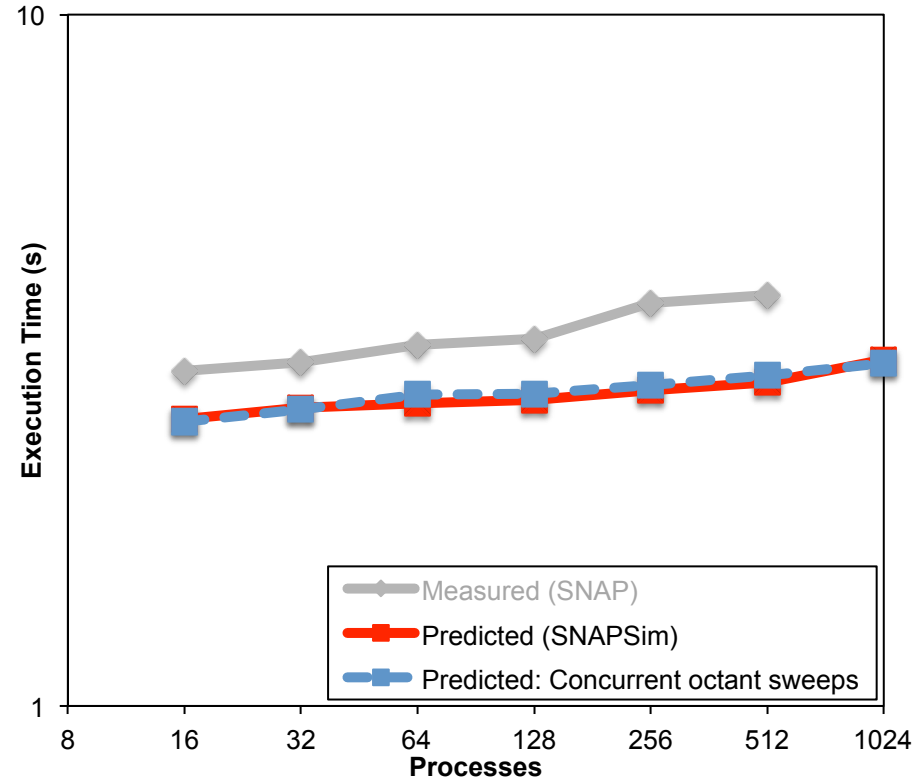
“Large” predicts better scaling, but ultimately captures same trend. Need to incorporate into MPI-decomposed simulator model and test.

Using SNAPSim to predict alternate sweep schedule absent present SNAP capability

Strong (cells=65k, nang=48, ng=42)



Weak (cells=16k/rank, nang=48, ng=42)



Does not reveal great benefit: need to test larger processor counts. Does reveal slight cost of overhead. Permits much faster exploration of scheduling choices.

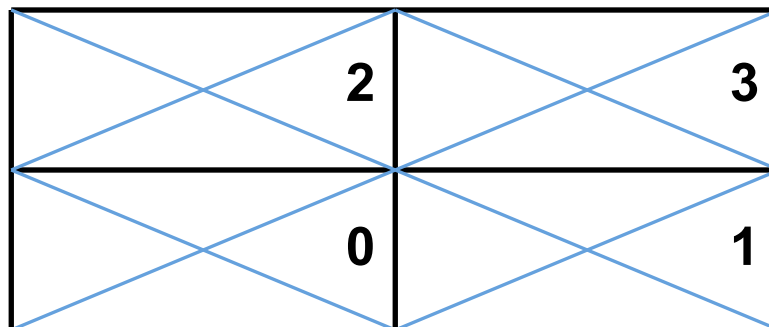
SNAPSim outlook

- Minor improvements for fidelity
 - Material mapping
 - Within group scattering operator
 - Global operations (broadcasts, reductions)
- Better use of hardware model improvements for predicting on-node data motion
- Decrease simulator runtime
 - Simulations can take too long to run
 - SNAPSim itself does not scale well → insufficient parallel work of simulator
 - Investigating options to improve runtime: Simplified communication model, use PyPy, etc.

Cell-wise Block-Jacobi: CWBJSim

Cell-Wise Block-Jacobi (CWBJ) sweeps

- At every iteration exchange incoming data from previous iteration across parallel spatial subdomains (chunks)
- No scheduling strategy (chunks executed simultaneously), just step through a chunk's mesh cells in numeric order
- Solve a $pAG \times pAG$ linear system for p nodes, A angles and G energy groups on every mesh cell (LU solver `dgetrf[s]`)
- Investigate tradeoff between reduced scheduling complexity and increased flop count per chunk on different architectures



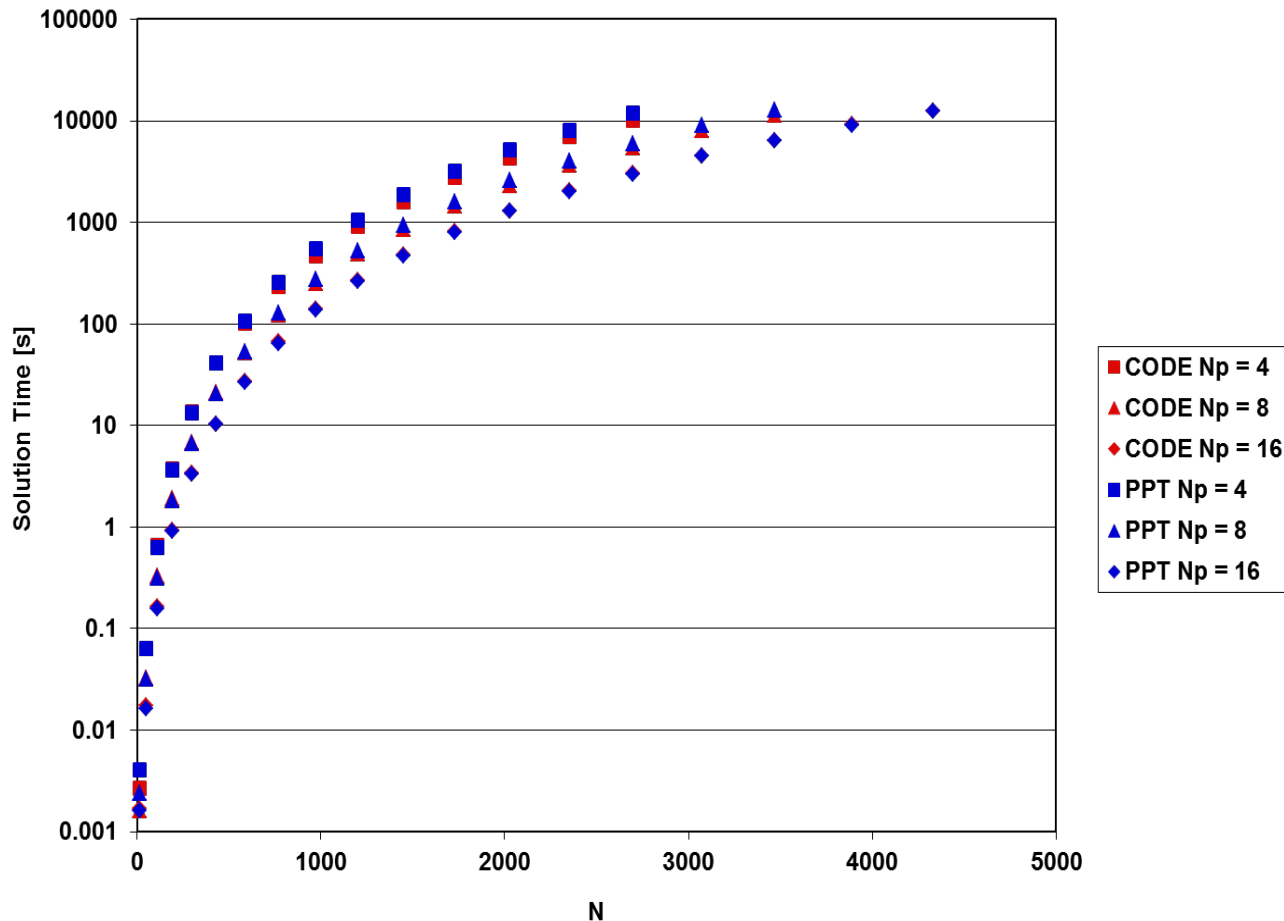
Chunk assigned to proc.

4 mesh cells per chunk

Validation of cwbjsim-mpi.py on Cielito

- $4 \times 8 \times 8 = 256$ triangular cells: $dx = dy = 10^{-4}$ cm
- Square Chebyshev-Legendre (CL) S_n : $n=2, \dots, 40$
- $g=1$, $c=0.99$, $\sigma=1 \text{ cm}^{-1}$, $q=1 \text{ m}^{-3}/\text{s}$
- $N=3 \times g \times n \times n$: dimension of phase space per triangle
- Vacuum BCs
- Iterative solver: GMRES(20), max it.=100, $\epsilon=10^{-5}$
- $N_p = 4, 8, 16$

Validation results on Cielito

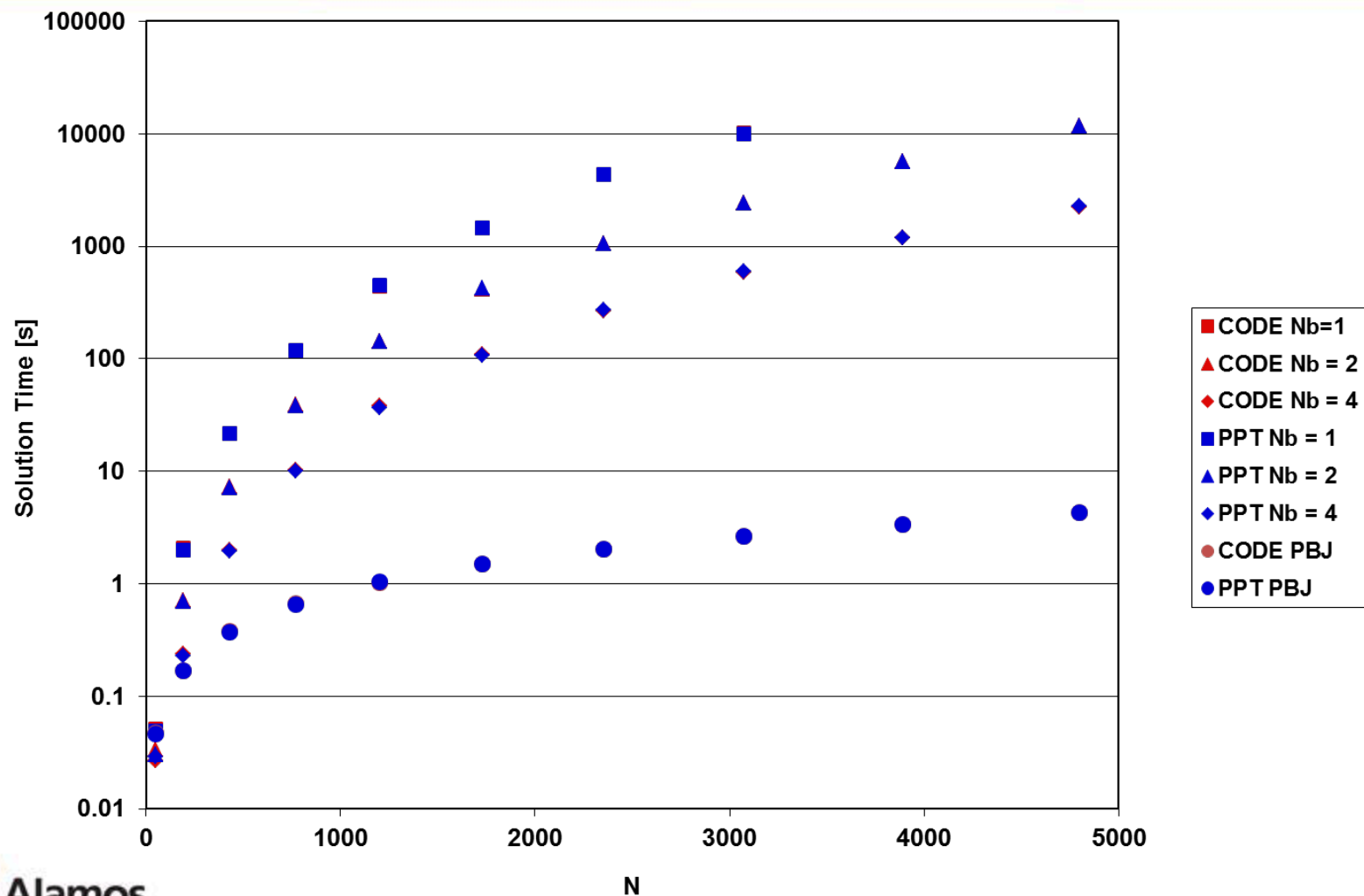


- float_alu_ops proportional to N^3 from literature for LU factorization appears to drive curves' trends at large N
- Code runs more optimally by spreading ranks across NUMA nodes at Np=4, 8
- PPT does not yet capture effect of NUMA boundaries on node → **OPEN QUESTION**

Algorithmic variations for CWBJ

- **Bundle group work**: local matrix solve decreases quadratically with number of groups
- **Parallel Block Jacobi**: all scattering operations on RHS, solve with local mesh sweeps
- Consider performance *per iteration*: convergence separate
- Return to previous problem
 - $g=4$
 - $N_p=1$ only
 - $N_b=1, 2, 4$: number of group bundles: $g_b=4, 2, 1$

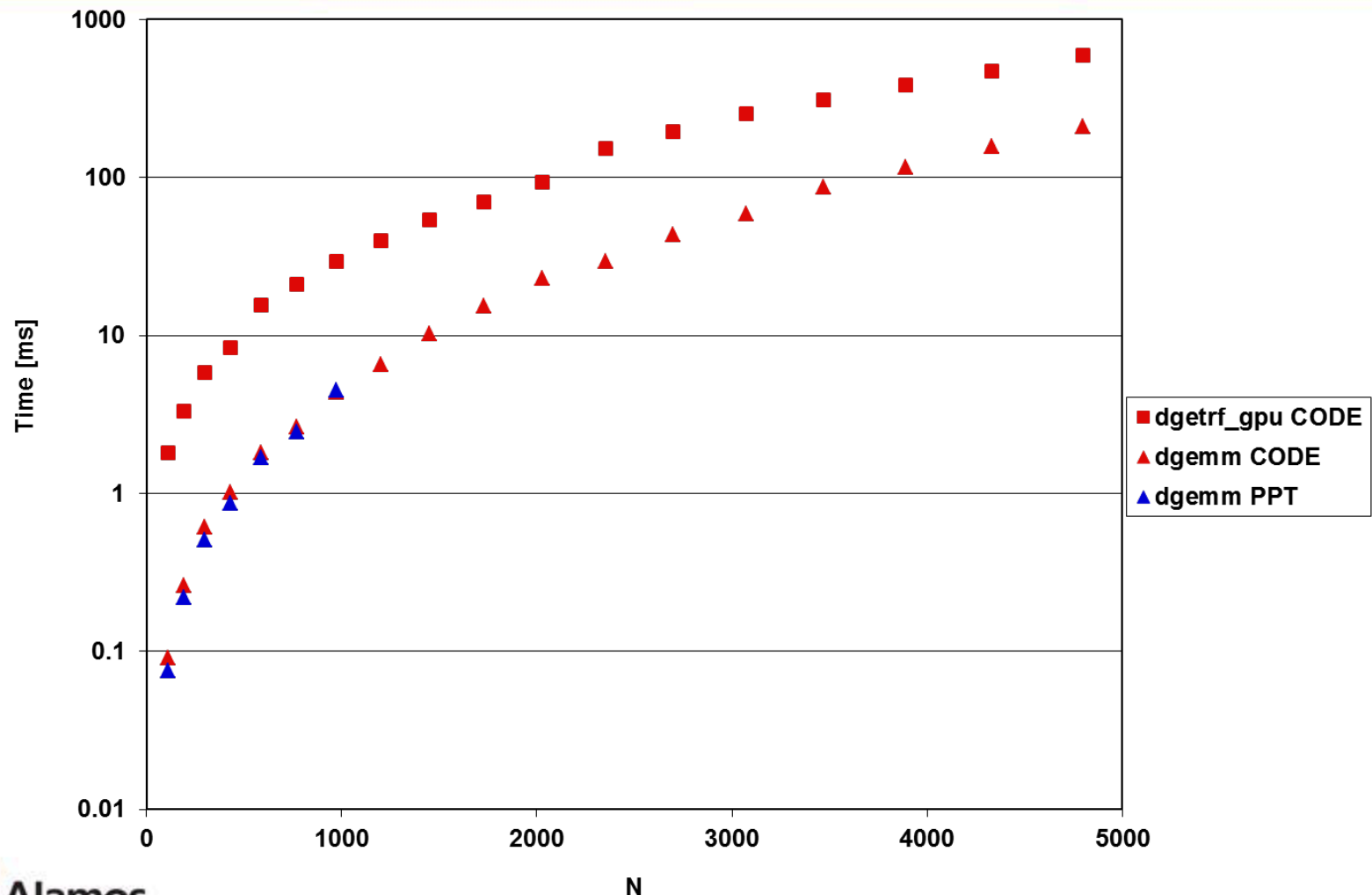
Serial results on Moonlight



Modeling CWBJ speedup on Moonlight GPUs

- Leveraging GPU hardware model originally developed by Guillaume Chapuis for a Titan node
- Added to performance prediction toolkit hardware model for a Moonlight node (Intel Host + NVIDIA M2090 GPU)
- Added to GPUPTest.py LU_APP test to simulate call to MAGMA dgetrf_gpu from GPU version of CWBJ
- To simplify generation of GPU task-list assume bulk of GPU work is in repeated call to dgemm_fermi.cu kernel
- For $n=6$ and $g=1$ initial case obtained good prediction (~ 0.08 ms) for actual kernel time (~ 0.09 ms) but overall dgetrf_gpu time appears to be higher (~ 1.8 ms)

Preliminary results for GPU model



CloverLeafSim

Eulerian hydro on a single-level mesh: CloverLeafSim

- Compressible Euler equations are used in a number of lab codes
- Finite volume formula common to a number of codes (Lagrangian step followed by advective remap)
- Explicit time integration is commonly used
- Fortran based (~4500 LoC)
- MPI, OpenMP, OpenACC, CUDA, OpenCL, PGAS

Validation against a 2D shock simulation

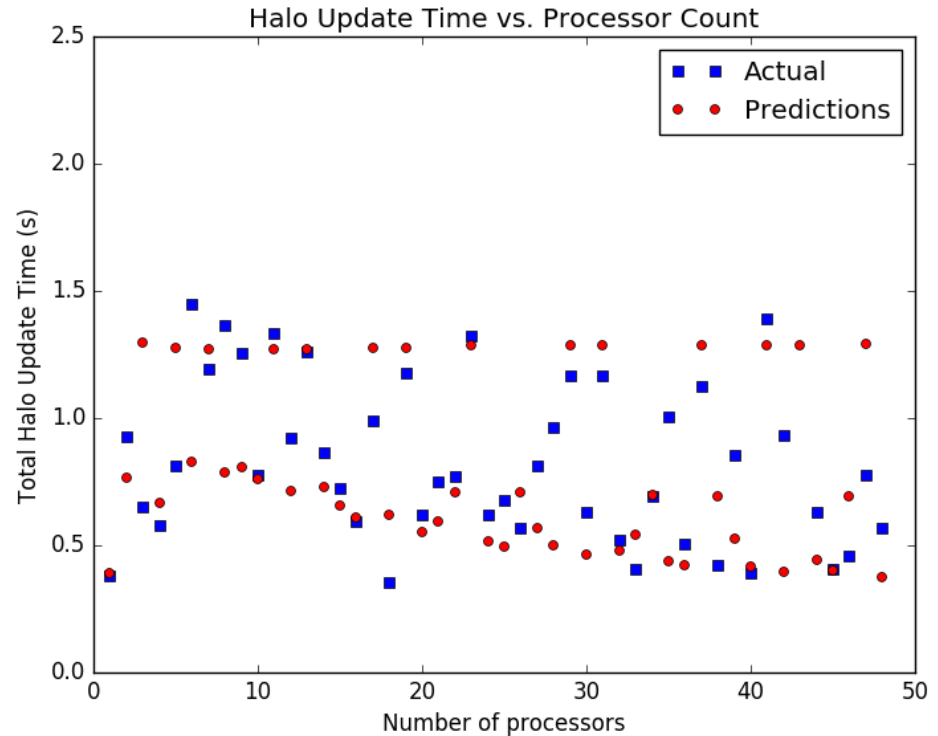
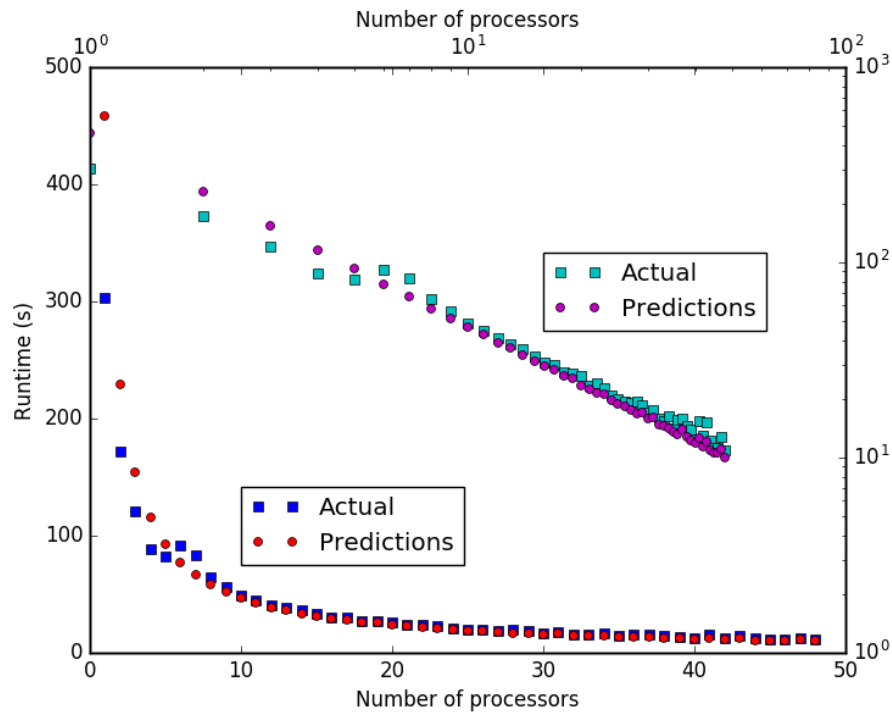



Table of Contents

- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
-  Middleware Models
- Hardware Models
- Parallel Discrete Event Simulation – Simian
- List of Publications, Presentations
- Outlook

Middleware Models

- MPI
 - Separate talk by Jason Liu (2 publications)
 - Fairly comprehensive capability
- OpenMP
 - Basic parallelized loop functionality implemented, used in SNAPSim
 - Additional functionality to be added on demand
- *Legion*
 - Postdoc hire with February start date will focus on a Legion model
- Interconnect models
 - Separate talk on Torus, FatTree, Dragonfly

MPI & Interconnect Models

- **Diverse models** of interconnection networks
 - All common topologies (torus, fat tree, dragonfly)
 - Existing production systems and academic abstractions
- **Easy integration** with applications
 - Stylized applications with focus on loop structures, important numerical kernels
 - Incorporate detailed MPI operations: communication patterns
- **Accurate and high-performance simulation** of communication behavior
 - Packet-level simulation (rather than phit-level simulation) provides sufficiently accurate results
 - Fully parallelizable models
 - Simian for just-in-time parallel discrete-event simulation

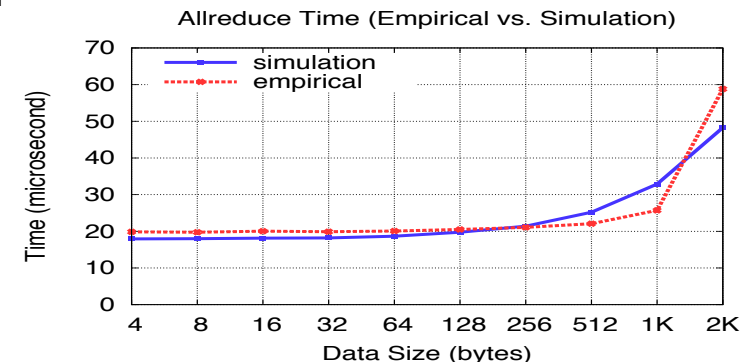
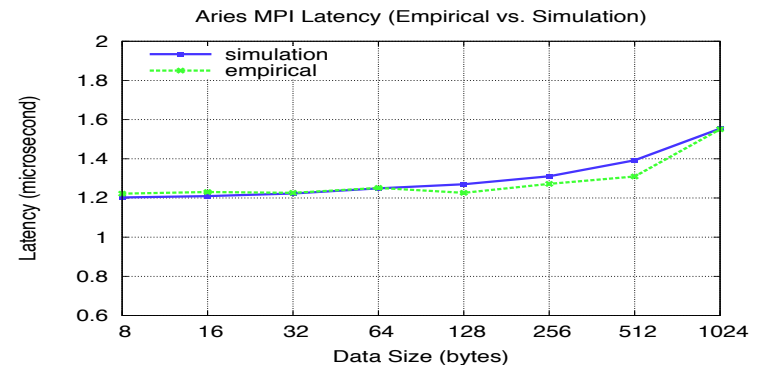
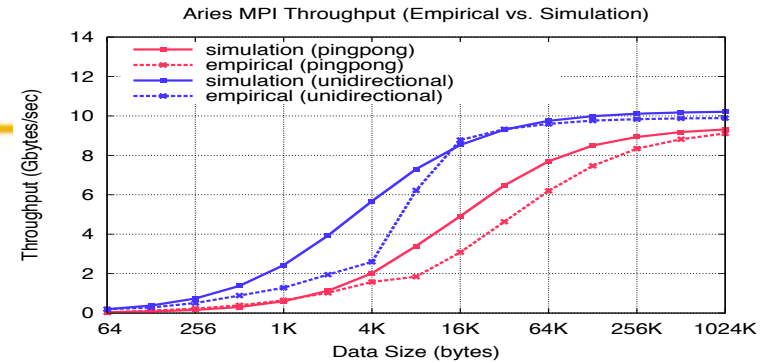



Table of Contents

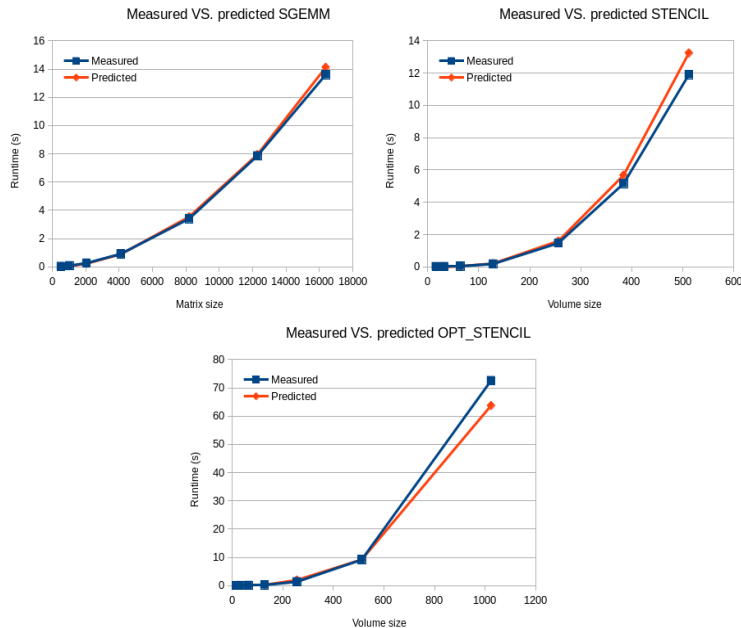
- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
- Middleware Models
-  Hardware Models
 - Parallel Discrete Event Simulation – Simian
 - List of Publications, Presentations
 - Outlook

(Compute Node) Hardware Models

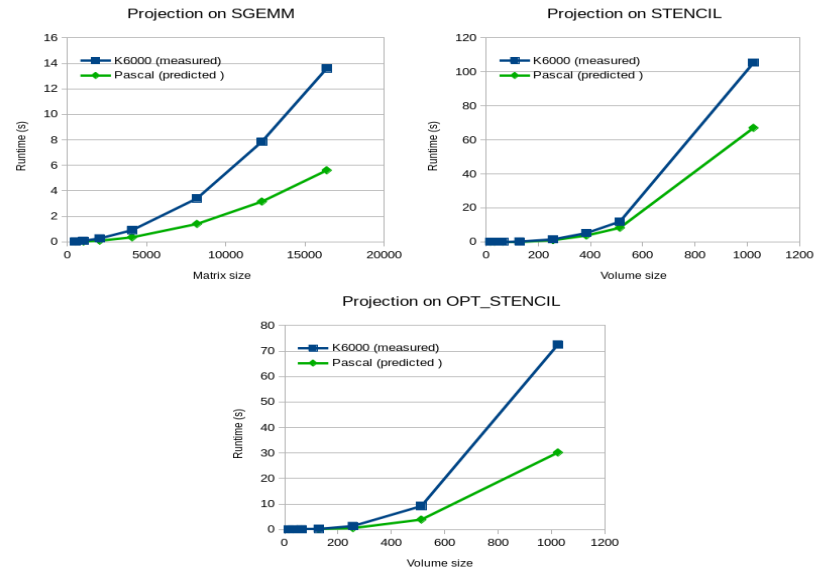
- Simple “First-Principles” parameter based models (“encode spec sheets”) with linear task lists
 - Works ok for simple architectures, such as GPU
 - Traditional CPU architectures: non-linearities due to memory hierarchies, pipelining, pre-fetching, and speculative execution. Low-level parallelism. Hard to find application-independent parameter values, albeit finding parameters for individual applications not too hard
- Learning-based hardware models
 - Training sets generated from benchmark application runs
 - Publication at ISC Workshop PMMA on energy use prediction
 - Large-scale effort stopped after several attempts
- Static basic block simulation on pipeline models with reuse distance computation and branch probability analysis has potential to be a game-changer
 - See talk by N. Santhi

Hardware Modeling: GPU Model Validation

Validation Results for K6000



Performance Prediction for Pascal



- ***Publication received Best Paper Award at ValueTools2016***
- Validation of three GPUs against Parboil benchmarks within 20%
- Performance Prediction of next generation Pascal GPU predicts speed-up of about 2.5 x
- Model is “first principles” values from spec sheet

Static Block Based Analysis, Memory, Pipeline Model

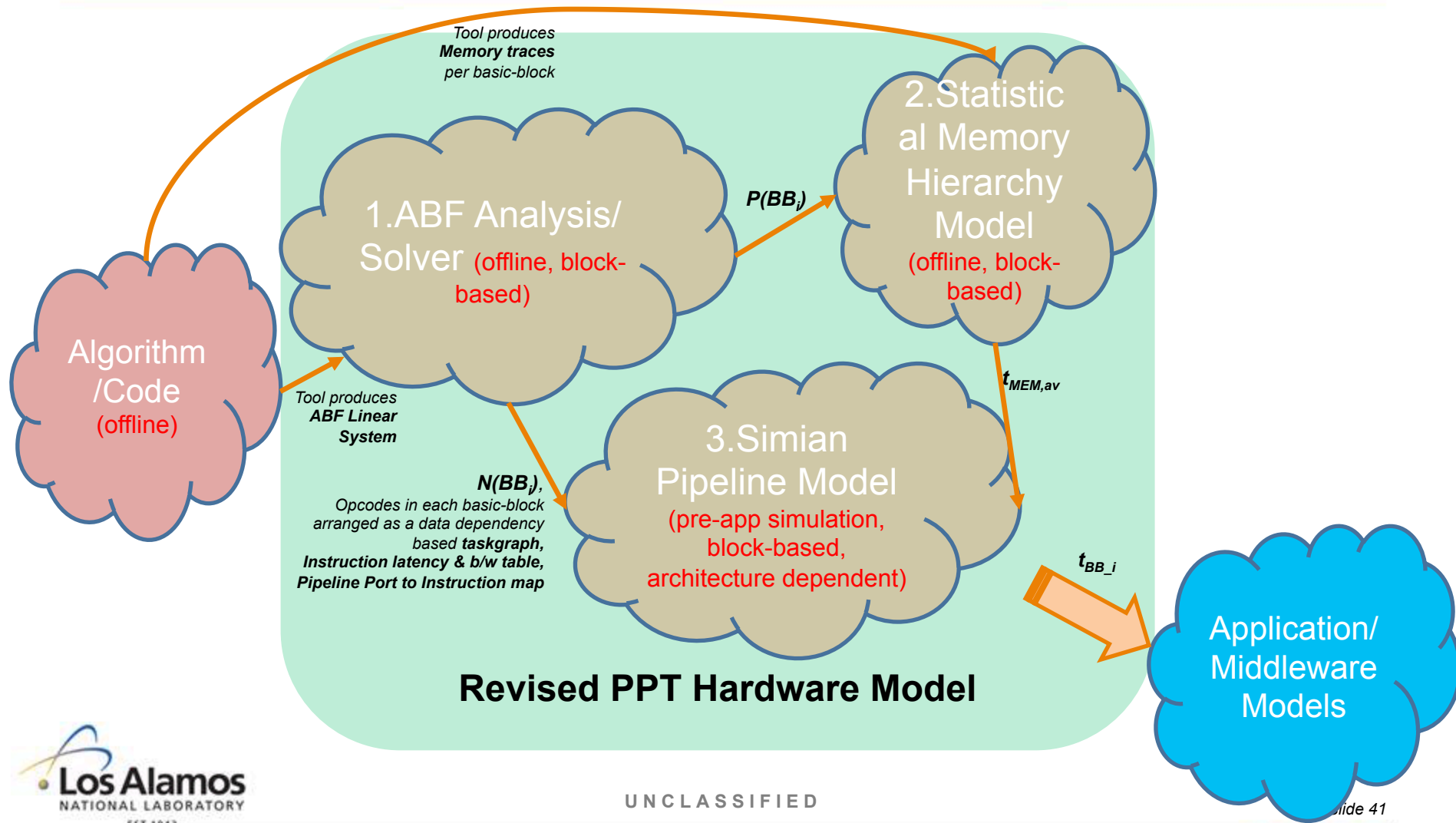
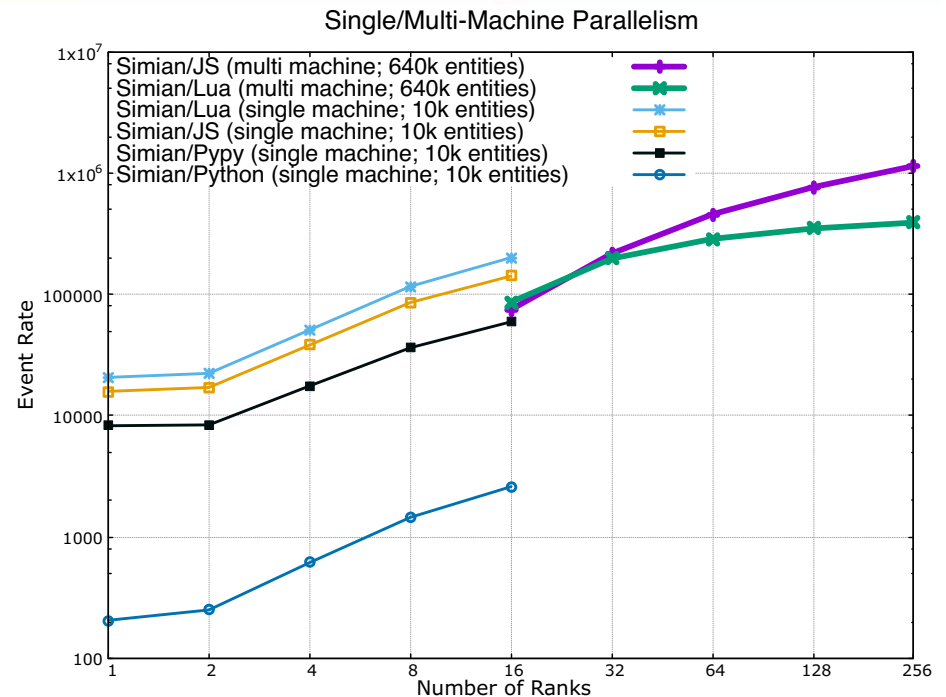


Table of Contents

- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
- Middleware Models
- Hardware Models
- ➔ Parallel Discrete Event Simulation – Simian
 - List of Publications, Presentations
 - Outlook

SimianJS under LA-PDES Benchmark Suite

- In barebones tests, SimianJS has reached event rates > 4 million events/s in serial mode on Mac desktops, which is 2X better than SimianLua
- **LA-PDES Benchmark Runs:** SimianLua initially performs better with MPI over multiple machines – due to the efficient, JITed C/FFI in LuaJit. At higher entity and rank counts, better memory management and garbage collection pulls SimianJS ahead.



Two scenarios are plotted: (a) Single machine: there are 10,000 entities in total for each simulation (b) Multi machine: there are 640,000 entities in total for each simulation. A sufficiently large number of entities ensures that strong scaling is apparent at higher rank counts.

SimianJS Architecture, MPI4JS

- SimianJS is implemented over a *LANL customized Mozilla Spidermonkey* open-source codebase
- MPI4JS interface was developed to support JS-native calls to MPI from usual stand-alone (not browser based) Javascript code
- Several other useful C functions are also exposed to the Javascript side
- As a PDES engine, SimianJS user-API closely mirrors both SimianLua and SimianPie. Use of MPI for simulation is optional.

User's PDES Model Code in Javascript

SimianJS Kernel


Spidermonkey JIT Engine

MPI4JS Layer (Optional)

MPI (MPICH/OpenMPI C/C++)

NW/OS/Hardware etc

Table of Contents

- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
- Middleware Models
- Hardware Models
- Parallel Discrete Event Simulation – Simian
-  List of Publications, Presentations
- Outlook

Publications 1/3

Journals

1. Richard J. Zamora, Arthur F. Voter, Danny Perez, Nandakishore Santhi, Susan M. Mniszewski, Sunil Thulasidasan, Stephan J. Eidenbenz:
Discrete event performance prediction of speculatively parallel temperature-accelerated dynamics.
Simulation 92(12): 1065-1086 (2016)
2. Guillaume Chapuis, Stephan Eidenbenz, Nandakishore Santhi:
GPU Performance Prediction Through Parallel Discrete Event Simulation and Common Sense. EAI Endorsed Trans. Ubiquitous Environments 3(10): e4 (2016)
3. TADSim: Discrete Event-based Performance Prediction for Temperature Accelerated Dynamics
S.M. Mniszewski, C. Junghans, A. Voter, D. Perez, S. Eidenbenz
ACM Transactions on Modeling and Computer Simulation (TOMACS), 2015, 25:3

Publications 2/3

Peer-reviewed Conference Proceedings

1. Guillaume Chapuis, David Nicholaeff, Stephan Eidenbenz, Robert S. Pavel:
Predicting Performance of Smoothed Particle Hydrodynamics Codes at Large Scales
Proceedings of the 2016 Winter Simulation Conference
2. Kishwar Ahmed, Jason Liu, Stephan Eidenbenz, Joe Zerr :
Scalable Interconnection Network Models for Rapid Performance Prediction of HPC Applications
Proceedings of the 18th IEEE International Conference on High Performance Computing and Communications (HPCC 2016)
3. Hristo Djidjev, Stephan Eidenbenz, B. Nadiga, EJ Park:
Simulation-Based and Analytical Models for Energy Use Prediction
Proceedings of 2nd International Workshop on Performance Modeling: Methods and Applications (PMMA16), at
ISC High Performance 2016, Frankfurt, Germany, June 23
4. Qiang Guan, Nathan BeBardleben, Panruo Wu, Stephan Eidenbenz, Sean Blanchard, Laura Monroe, Elisabeth Baseman, and Li Tan. 2016.
Design, Use and Evaluation of P-FSEFI: A Parallel Soft Error Fault Injection Framework for Emulating Soft Errors in Parallel Applications.
In Proceedings of the 9th EAI International Conference on Simulation Tools and Techniques (SIMUTOOLS'16).
ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, 9-17.
5. Kishwar Ahmed, Mohammad Obaida, Jason Liu, Stephan Eidenbenz, Nandakishore Santhi, Guillaume Chapuis:
An Integrated Interconnection Network Model for Large-Scale Performance Prediction.
ACM SIGSIM-PADS 2016: 177-187

Publications 3/3

Peer-reviewed Conference Proceedings

6. N. Prajapati, W. Ranasinghe, S. Rajopadhye, R. Andonov, H. Djidjev, and T. Grosser:
Simple, Accurate, Analytical Time Modeling and Optimal Tile Size Selection for GPGPU Stencils
22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), 2017.
7. Joan Boyar, Stephan J. Eidenbenz, Lene M. Favrholt, Michal Kotrbčik, Kim S. Larsen:
Online Dominating Set
Proceedings of SWAT 2016: 21:1-21:15, also available as arXiv preprint arXiv:1604.05172
8. Guillaume Chapuis, Stephan Eidenbenz, Nandakishore Santhi:
GPU Performance Prediction Through Parallel Discrete Event Simulation and Common Sense
Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS 2015), pp 204 -2011, 2015
9. Guillaume Chapuis, Stephan Eidenbenz, Nandakishore Santhi, Eun Jung Park:
Simian integrated framework for parallel discrete event simulation on GPUs.
Winter Simulation Conference 2015: 1127-1138
10. Eunjung Park, Stephan Eidenbenz, Nandakishore Santhi, Guillaume Chapuis, Bradley W. Settlemyer:
Parameterized benchmarking of parallel discrete event simulation systems: communication, computation, and memory.
Winter Simulation Conference 2015: 2836-2847
11. Nandakishore Santhi, Stephan Eidenbenz, Jason Liu:
The simian concept: parallel discrete event simulation with interpreted languages and just-in-time compilation.
Winter Simulation Conference 2015: 3013-3024

Presentations

1. NECDC 2016 at LANL: SNAPSim, Joe Zerr, February 2016
2. CPAM Review at LANL: SNAPSim, Joe Zerr, April 2016
3. CPAM Review at LANL: Simian PDES Engine, N. Santhi, April 2016
4. CPAM Review at LANL: PPT GPU Model, G. Chapuis, April 2016
5. Salishan Conference (Oregon): Performance Prediction Toolkit, Jason Liu, April 2016
6. ModSim Workshop (PNNL): Performance Prediction Toolkit, Jason Liu, July 2016
7. JOWOG-34: PPT overview, S. Eidenbenz, February 2017

+ conference proceedings presentations

Software

1. Simian – A Parallel Discrete Event Simulation Engine for Interpreted Languages and Just-in-time Compilation
 - Open-sourced December 2015
 - Includes Python, Lua, and Javascript versions
 - Includes LA-PDES Benchmark app (developed as part of this DR)
2. mpi4js – An MPI Interface for Javascript
 - To be open-sourced Spring 2017
3. Performance Prediction Toolkit - PPT
 - Open-sourcing planned after stabilization, in particular on hardware model side

Table of Contents

- Introduction: Performance Prediction Toolkit (PPT)
- Application Models
- Middleware Models
- Hardware Models
- Parallel Discrete Event Simulation – Simian
- List of Publications, Presentations

 Outlook

Outlook 1/3

- Technical Goals
 - Improve quality of memory hierarchy, pipelining, prefetching models
 - Resort to cycle-level simulation for (static) basic blocks
 - Improve scalability of prediction capability
 - Communication model drives event count at low MinDelays
 - Moving away from PDES-only approach, coupling with analytics
 - More aggressive use of application-specific speed-up tricks
 - Use of MPI/Interconnect model only when clearly needed
 - Learning scaling behavior at smaller rank-count
 - Increased use of pypy (Just-in-time Compilation for Python)
 - Lua or JS-based implementations (1-2 magnitude improvement)
 - Add Legion as Middleware model
 - Code clean-up and open-sourcing

Outlook 2/3

- Novel Computing Performance Prediction
 - Leverage of FY16 (reserve) investments in ASIC design for Molecular Dynamics and other mission-relevant applications has been taken over by ASC's Beyond Moore's Law thrust at 100k funding level
 - FY16 Familiarization with DWave Quantum Computing
 - Potential predictive capability for embedding algorithms onto Dwave's Chimera graph

Outlook 3/3

- Outreach Goals
 - Integration into work flows of multiple ECP projects
 - Codesign centers, ATDM
 - Early NSCI/ECP adopters
 - Molecular Dynamics (A. Voter)
 - Beyond Moore's Law (N. Santhi)
 - Adoption by PARTISN, IMC teams
 - Other suggestions?

APPENDIX

Performance Prediction Team FY15/16

Stephan Eidenbenz (PI)	NSEC	Balu Nadiga	CCS-2
Joe Zerr (co-PI)	CCS-2	Gabe Rockefeller	CCS-2
		Chris Fryer	CCS-2
Nandkishore Santhi	CCS-3	Randy Baker	CCS-2
Eun Jung Park (PD)	CCS-3	Max Rosa	CCS-2
Guillaume Chapuis (PD)	CCS-3	Kris Garrett (PD)	CCS-2
Sunil Thulasidasan	CCS-3		
Hristo Djidjev	CCS-3	Phil Romero	HPC-1
Patrick Kelly	CCS-3	Mike Warren	T-2
Ben Reidys (HS)	CCS-3	Steve Nolen	XCP-3
		Qiang Guan (PD)	HPC-5/CCS-7
Jason Liu	Florida Intl U	Rick Zamora (PD)	T-1
Mohammad Obaida (GRA)	Florida Intl U		
Ahmed Kishwar (GRA)	Florida Intl U		

Performance Prediction Toolkit (PPT):

Rapid Prototyping Modeling (Python or Lua): Simple, Modular

Code

Hardware Model Library

- Clusters
Mustang, Trinity, Cielo, Titan
- Nodes, Cores
AMD Opteron, KNL, MacPro
- Accelerators
K20X, K40, K6000, M2090, Pascal
- Interconnect
Gemini 3D Torus MPI
- File system (Lustre)

Application Simulator Library

- Benchmark Apps
- PolyBenchSim
 - ParboilSim
- Production Apps
- SNAPSim
 - SPHSim
 - SpecTADSim

Simian – Parallel Discrete Simulation Engine

Data

Learned Time Functions



Application Instrumentation Data

PolyBench, SNAP, SPH, CloverLeaf

Hardware Specs Data

Mustang, Haswell, IvyBridge, SandyBridge, Vortex

Modeling the Computing Stack: Select Level of HW Detail based on Suspected Bottleneck Resources

Hardware -> <i>Entities</i>	Software -> <i>Processes</i>	
Cluster	Equation, Method	 Hardware architecture-independent software specification
Nodes, Interconnect/Network	Algorithm	
Cores, (Main) Memory	High-level Language (C, Python, Fortran)	
HW Threads, ALU, Vector units	Intermediate Representation (IR)	
Registers	Machine Code, Assembly	
Memory Cache Levels	Prefetching, Purging	 Processes implemented in hardware
Pipelines	Pipelining, Speculative execution	
...	...	

Codesign Performance Modeling Paradigm

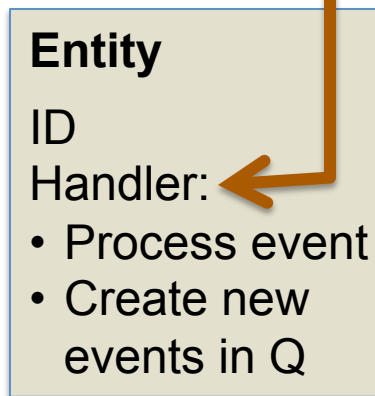
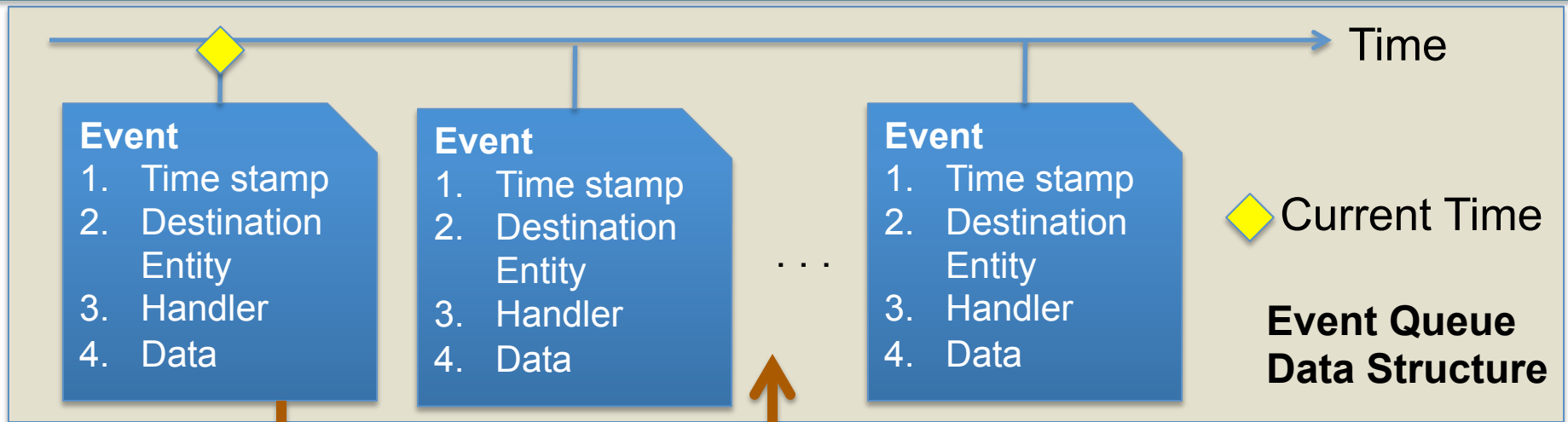
- *Hardware* = *Entities* at a chosen level of detail that could be bottleneck resources
- *Software* = *Processes* running on entities
- Processes *interact* with and *wait* for other processes on different (or same) entities
- Processes *advance their local time* (“sleep”) to mimic computation or other resource usage not modeled in more detail

Performance Prediction Toolkit - Repository

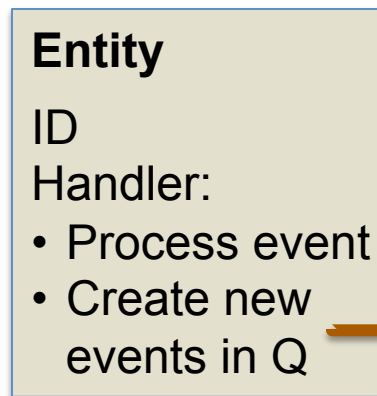
The screenshot shows the GitLab web interface for the 'performance_prediction_toolkit' repository. The browser address bar displays 'https://gitlab.lanl.gov/pp_codesign/performance_prediction_toolkit/tree/master/code'. The repository name 'pp_codesign / performance_prediction_toolkit' is shown at the top, along with a search bar and navigation icons. Below the repository name, tabs for 'Project', 'Activity', 'Repository', 'Pipelines', 'Graphs', 'Issues', 'Merge Requests', and 'Wiki' are visible. The 'Repository' tab is active, showing a file browser view. The file browser shows the 'master' branch and the path 'performance_prediction_toolkit / code /'. A table lists the directory structure and commit history:

Name	Last Commit > f7b1e97a – about a month ago latest version History	Last Update
..		
apps	latest version	about a month ago
hardware	Adding MLIntelPlusGPUCore to processors and MLIntelPlusGPUNode to nodes. Adding fir...	about a month ago
middleware	updated mpi and intercon partitioning.	2 months ago
simian	updated mpi and intercon partitioning.	2 months ago

Discrete Event Simulation (DES) Processes Events of a Simulated System at Discrete Time Steps



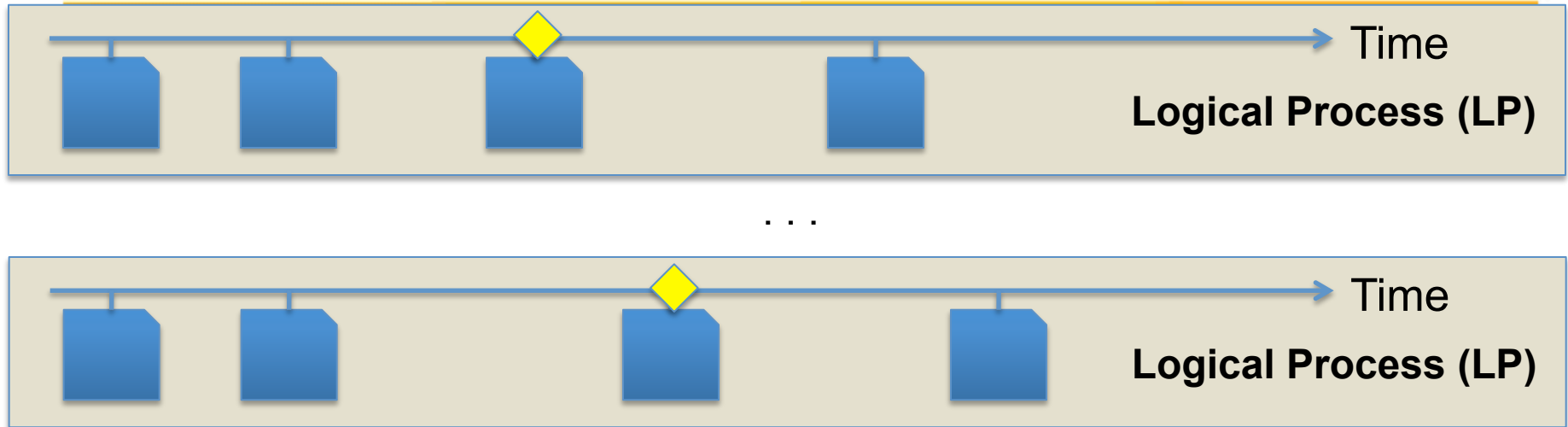
...



Main Loop:

While Event Queue not empty
Pop next event;
Advance current time;
Call Handler function
on destination entity;

Parallel Discrete Event Simulation (PDES) Synchronizes Multiple Time Lines



- The local current times of LPs synchronized through MPI_All_Reduce calls
- Entities live on one LP; handlers can create events with destination entities on other LPs through MPI_send calls
- Cross LP-events get time stamps at least MinDelay into future

Related Work: HPC Simulations

- Full system simulators:
 - Simics, SimpleScalar, GEM5, COTSon, PTLsim, Asim
- Analytical tools:
 - TAU, Vampir, HPCToolkit, Paraver, PACE, ASPEN, Palm, GROPHECY
- Processor/core simulators:
 - McSimA+, Zsim, Manifold
- Memory system simulators (DRAM, NVM, Cache):
 - DRAMSim, USIMM, DrSim, Ramulator, NVMain
- NoC simulators:
 - BookSim, GARNET, DARSIM, HORNET, TOPAZ, DNOC
- FPGA-based simulators:
 - Ramp Gold, HAsim, DART, Arete

Related Work: Performance Prediction

- **PPT (LANL):** physics-application oriented suite of performance prediction models for large-scale systems. Python, Lua. Includes hardware interconnect and node-level models, MPI and OpenMP, and a set of parameterized physics codes models
- **BigSim (UIUC):** for performance prediction of large-scale parallel machines (with relatively simple interconnect models), implemented in Charm++ and MPI, shown to scale up to 64K ranks
- **xSim (ORNL):** scale to 128M MPI ranks using PDES with lightweight threads, include various interconnect topologies (high-level models, e.g., network congestion omitted)
- **SST and SST Macro (SNL):** a comprehensive simulation framework, separate implementation, one intended with cycle-level accuracy and the other at coarser level for scale
- **CODES (ANL):** focused on storage systems, built on ROSS using reverse computation simulation that scales well